INTERNET-DRAFT
draft-ietf-cat-kerberos-pk-cross-05.txt
Updates: RFC 1510
expires May 15, 1999

Brian Tung
Tatyana Ryutov
Clifford Neuman
Gene Tsudik
ISI
Bill Sommerfeld
Hewlett-Packard
Ari Medvinsky
Matthew Hur
CyberSafe Corporation

Public Key Cryptography for Cross-Realm Authentication in Kerberos

0. Status Of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ftp.ietf.org (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

The distribution of this memo is unlimited. It is filed as draft-ietf-cat-kerberos-pk-cross-05.txt, and expires May 15, 1999. Please send comments to the authors.

1. Abstract

This document defines extensions to the Kerberos protocol specification (RFC 1510, 'The Kerberos Network Authentication Service (V5)', September 1993) to provide a method for using public key cryptography during cross-realm authentication. The methods defined here specify the way in which message exchanges are to be used to transport cross-realm secret keys protected by encryption under public keys certified as belonging to KDCs.

2. Introduction

The advantages provided by public key cryptography have produced a demand for use by the Kerberos authentication protocol. A draft describing the use of public key cryptography in the initial authentication exchange in Kerberos has already been submitted. This draft describes its use in cross-realm authentication.

The principal advantage provided by public key cryptography in cross-realm authentication lies in the ability to leverage the existing public key infrastructure. It frees the Kerberos realm administrator from having to maintain separate keys for each other realm with which it wishes to exchange authentication information, or from having to utilize a hierarchical arrangement, which may pose problems of trust.

Even with the multi-hop cross-realm authentication, there must be some way to locate the path by which separate realms are to be transited. The current method, which makes use of the DNS-like realm names typical to Kerberos, requires trust of the intermediate KDCs.

The methods described in this draft allow a realm to specify, at the time of authentication, which certification paths it will trust. A shared key for cross-realm authentication can be established for a set period of time. Furthermore, these methods are transparent to the client; therefore, only the KDCs need to be modified to use them.

It is not necessary to implement the changes described in the "Public Key Cryptography for Initial Authentication" draft in user clients in order to make use of the changes in this draft. We solicit comments about the interaction between the two protocol changes, but as of this writing, the authors do not perceive any obstacles to using both.

3. Protocol Specification

We assume that the client has already obtained a TGT. To perform cross-realm authentication, the client does exactly what it does with ordinary (i.e., non-public-key-enabled) Kerberos; the only changes are in the KDC; although the ticket which the client forwards to the remote realm may be changed. This is acceptable since the client treats the ticket as opaque.

The revised PKCROSS protocol makes use of a proposed field in the Kerberos response, namely a TicketExtension. This field is not part of the encrypted part of the ticket (although it may have been encrypted earlier), but it accompanies the ticket and should be passed along by unaware clients with the rest of the ticket in an opaque fashion. (However, some incompletely-implemented ASN.1 compilers/parsers may fail when the TicketExtension is used.) Using this field allows us to achieve the following objectives:

* Avoid modification of clients and application servers.

* Allow remote KDC to control its policy on cross-realm keys shared between KDCs, and on cross-realm tickets presented by clients.

* Remove any need for KDCs to maintain state about keys shared with other KDCs.

* Leverage work already done for PKINIT.

The PKCROSS protocol does not preclude the use of "short-cut" strategies, in which clients may take advantage of advance knowledge of shorter transited chains.

3.1. Overview of Protocol Changes

The basic operation of the revised PKCROSS protocol is as follows:

1. The client submits a request to the local KDC for credentials for the remote realm.

2. The local KDC submits a PKINIT request to the remote KDC. This request has a flag set to indicate that the request is a special cross-realm key request.

3. The remote KDC responds as per PKINIT, except that the ticket contains a TicketExtension which indicates that this is a cross-realm key response. The TicketExtension may also contain policy information, which the local KDC must reflect in the credentials it forwards to the client. Call this ticket TGT_{L,R}.

4. The local KDC passes a ticket, TGT_{C,R}, to the client. This ticket contains in its TicketExtension field the cross-realm key ticket, TGT_{L,R}. This ticket is encrypted using the key sealed in TGT_{L,R}. (The TicketExtension field is not encrypted.) The local KDC may optionally include another TicketExtension type that indicates the hostname and/or IP address for the remote KDC.

5. The client submits the request directly to the remote KDC, as before.

Sections 3.2 through 3.4 describe in detail steps 2 through 4 above. Section 3.5 describes the conditions under which steps 2 and 3 may be skipped.

A comment about the KDC-to-KDC communication present in these protocol changes. If there is no such exchange between the KDCs, then the local KDC will have to issue a ticket with the expectation that the remote KDC will accept it, but no guarantee. This is a simple trade-off: non-repudiability of the ticket, so to speak, or avoiding KDC-to-KDC communication. As a matter of principle, we choose the former, so that operation from the client's perspective is unchanged.

3.2. Local KDC's PKINIT Request to Remote KDC

When the local KDC receives a request for cross-realm authentication, it sends a request to the remote KDC for a ticket, denoted by TGT_{L,R}. This request is in fact a PKINIT request as described in Section 3.2 of the PKINIT draft; i.e., it consists of an AS-REQ, with a PA-PK-AS-REQ included as a preauthentication field.

In addition, the local KDC indicates that this request is for a PKCROSS cross-realm key, by setting bit 9 in the kdc-options field of the AS-REQ. We propose to call this bit the PKCROSSKEY flag. Otherwise, this exchange exactly follows the description given by Section 3.2 of the PKINIT draft.

3.3. Remote KDC's PKINIT Response to Local KDC

When the remote KDC receives the PKINIT/CROSS request from the local KDC, it sends back a PKINIT response as described in Section 3.2 of the PKINIT draft, with the following changes: The remote KDC does not encrypt the encryptedTktPart with a random key, as described in the PKINIT draft, but instead encrypts it with a special symmetric key it uses for validating PKCROSS requests. This key, instead of a random key, is then placed in an envelope as described in the PKINIT draft.

In addition, a TicketExtension field of type 2 (TE-TYPE-PKCROSS) is included with the ticket (as a non-encrypted part):

```
TicketExtension ::= SEQUENCE OF SEQUENCE {
        te-type[0]       INTEGER,
        te-data[1]       OCTET STRING
} -- per the revised Kerberos specification
```

```
where
   te-type equals 2 for TE-TYPE-PKCROSS-KDC
and
   te-data is ASN.1 encoding of CrossRealmTktData.
```

```
CrossRealmTktData ::= SEQUENCE OF SEQUENCE {
        type                    [0] INTEGER,
        data                    [1] OCTET STRING
}
```

where types and the corresponding data are to be interpreted as follows:

type	interpretation	data is ASN.1 encoding of
1	lifetime (in seconds)	INTEGER

Further types may be added to this table.

3.4. Local KDC's Response to Client

Upon receipt of the PKINIT/CROSS response from the remote KDC, the local KDC formulates a response to the client. This reply is constructed exactly as in the Kerberos specification, except for the following:

* The local KDC places TGT_{L,R} in the TicketExtension field of the client's ticket for the remote realm (denoted here by TGT_{C,R}).
* The local KDC adds the name of its CA to the transited field of TGT_(C,R).

```
TicketExtension ::= SEQUENCE OF SEQUENCE {
        te-type[0]       INTEGER,
        te-data[1]       OCTET STRING
} -- per the revised Kerberos specification
```

```
where
   te-type equals 3 for TE-TYPE-PKCROSS-CLIENT
```

and
te-data is ASN.1 encoding of TGT_(L,R).

Optionally, the local KDC may add another TicketExtention where
te-type equals 5 for TE-TYPE-DEST-HOST
and
te-data is ASN.1 encoding of DestHost.

A client could then rely on the local KDC to provide a referral to the remote KDC, thus removing the need for the client to maintain local host/realm mapping information.

```
DestHost ::= SEQUENCE {
       DNSHostname [0]     OCTET STRING,
                           -- authoritative DNS hostname
       HostAddr    [1]     HostAddress OPTIONAL
                           -- as defined by RFC 1510
}
```

3.5. Short-Circuiting the KDC-to-KDC Exchange

Under certain circumstances, the KDC-to-KDC exchange described in Sections 3.2 and 3.3 may be skipped. The local KDC has a known lifetime for TGT_{C,R} (which may in part be determined by the policy sent along with TGT_{L,R}). If the local KDC already has a ticket TGT_{L,R}, and the start time plus the lifetime for TGT_{C,R} does not exceed the expiration time for TGT_{L,R}, then the local KDC may skip the exchange with the remote KDC, and issue a cross-realm ticket to the client as described in Section 3.4.

Since the remote KDC may change the special cross-realm symmetric key (referred to in Section 3.2) while there are cross-realm key tickets (the TGT_{L,R}) still active, it is obligated to cache those tickets until they expire.

4. Transport Issues

Because the messages between the KDCs involve PKINIT exchanges, and PKINIT recommends TCP as a transport mechanism (due to the length of the messages and the likelihood that they will fragment), the same recommendation for TCP applies to PKCROSS as well.

5. Expiration Date

This Internet-Draft will expire on May 15, 1999.

6. Authors' Addresses

Brian Tung
Tatyana Ryutov
Clifford Neuman
Gene Tsudik
USC/Information Sciences Institute
4676 Admiralty Way Suite 1001
Marina del Rey, CA 90292-6695

Phone: +1 310 822 1511
E-Mail: {brian, tryutov, bcn, gts}@isi.edu

Bill Sommerfeld
Hewlett Packard
300 Apollo Drive
Chelmsford MA 01824
Phone: +1 508 436 4352
E-Mail: sommerfeld@apollo.hp.com

Ari Medvinsky
Matthew Hur
CyberSafe Corporation
1605 NW Sammamish Road Suite 310
Issaquah WA 98027-5378
Phone: +1 206 391 6000
E-mail: {ari.medvinsky, matt.hur}@cybersafe.com